YOLO Object Detection Explained

Understand YOLO object detection, its benefits, how it has evolved over the years, and some real-life applications.

Contents

Updated Sep 28, 2024 · 22 min read

Zoumana Keita
A data scientist who likes to write and share knowledge with the data and IA community

TOPICS

Data Science

Object detection is a computer vision technique for identifying and localizing objects within an image or a video.

Image localization is the process of identifying the correct location of one or multiple objects using bounding boxes, which correspond to rectangular shapes around the objects. This process is sometimes confused with image classification or image recognition, which aims to predict the class of an image or an object within an image into one of the categories or classes.

The illustration below corresponds to the visual representation of the previous explanation. The object detected within the image is a “Person.”

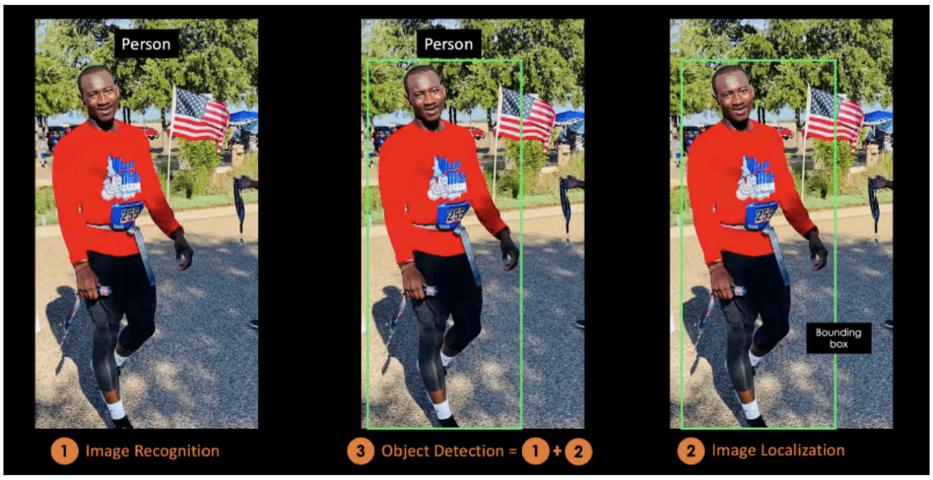


Image by Author

In this conceptual blog, you will first understand the benefits of object detection before introducing YOLO, the state-of-the-art object detection algorithm.

In the second part, we will focus more on the YOLO algorithm and how it works. After that, we will provide some real-life applications using YOLO.

The last section will explain how YOLO evolved from 2015 to 2024 before concluding on the next steps.

What is YOLO?

You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their famous research paper You Only Look Once: Unified, Real-Time Object Detection.

The authors frame the object detection problem as a regression rather than a classification task by spatially separating bounding boxes and associating probabilities to each detected image using a single convolutional neural network (CNN).

If you're interested in image classification, consider taking the Image Processing with Keras in Python course, where you'll build Keras-based deep neural networks for image classification tasks. If you are more interested in Pytorch, Deep Learning with Pytorch will teach you about convolutional neural networks and how to use them to build much more powerful models.

What Makes YOLO Popular for Object Detection?

Some of the reasons why YOLO is leading the competition include its:

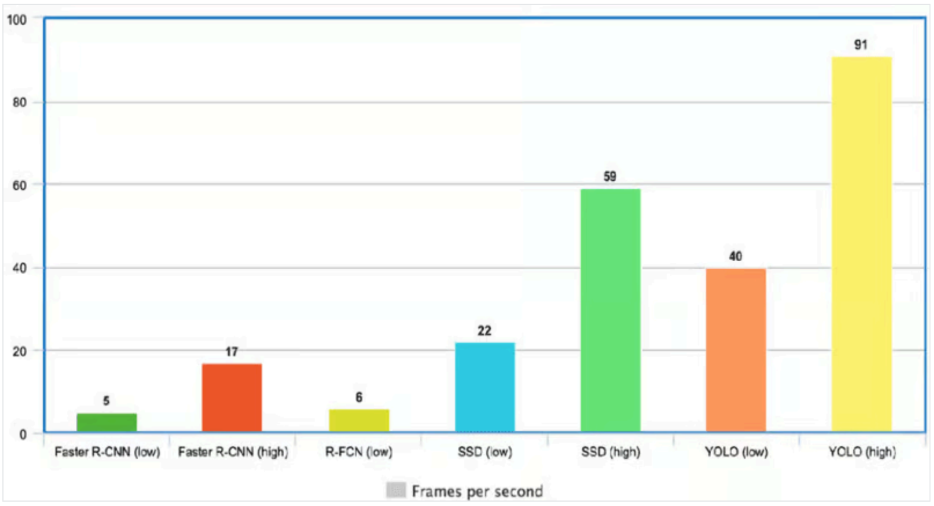
- Speed
- Detection accuracy
- Good generalization
- Open-source

Let's see these features in more detail.

1. Speed

YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing.

From the graphic below, we observe that YOLO is far beyond the other object detectors with 91 FPS.



YOLO speed compared to other state-of-the-art object detectors (source)

2. High detection accuracy

YOLO is far beyond other state-of-the-art models in accuracy, with very few background errors.

3. Better generalization

This is especially true for the new versions of YOLO, which will be discussed later in the article. With those advancements, YOLO has gone a little further by providing better generalization for new domains, which makes it great for applications relying on fast and robust object detection.

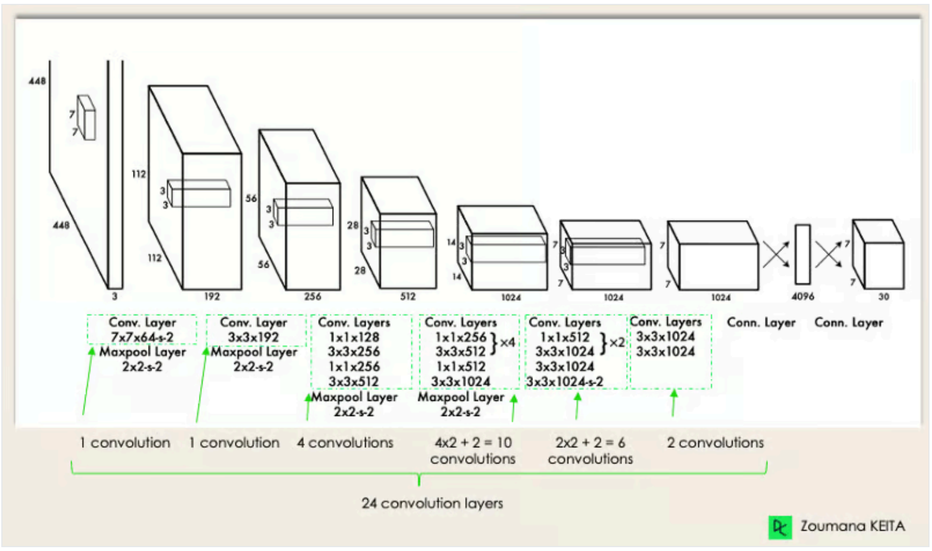
For instance, the Automatic Detection of Melanoma with Yolo Deep Convolutional Neural Networks paper shows that the first version, YOLOv1, has the lowest mAP for the automatic detection of melanoma disease, compared to YOLOv2 and YOLOv3.

4. Open-source

Making YOLO open-source has led the community to improve the model constantly. This is one of the reasons why YOLO has made so many improvements in such a limited time.

YOLO Architecture

YOLO architecture is similar to **GoogleNet**. As illustrated below, it has 24 convolutional layers, four max-pooling layers, and two fully connected layers.


448
448
7
7
3
112
112
3
3
192
56
56
3
3
256
28
28
3
3
512
14
14
3
3
1024
7
7
3
3
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
1 convolution
1 convolution
4 convolutions
4x2 + 2 = 10 convolutions
2x2 + 2 = 6 convolutions
2 convolutions
24 convolution layers
Zoumana KEITA

YOLO Architecture from the **original paper** (Modified by Author)

The architecture works as follows:

- Resizes the input image into 448×448 before going through the convolutional network.
- A 1×1 convolution is first applied to reduce the number of channels, followed by a 3×3 convolution to generate a cuboidal output.
- The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.
- Some additional techniques, such as batch normalization and dropout, regularize the model and prevent it from overfitting.

By completing the **Deep Learning in Python** course, you will be ready to use Keras to train and test complex, multi-output networks and dive deeper into deep learning.

How Does YOLO Object Detection Work?

Now that you understand the architecture let’s take a high-level overview of how the YOLO algorithm performs object detection using a simple use case.

“Imagine you built a YOLO application that detects players and soccer balls from a given image.

But how can you explain this process to someone, especially non-initiated people?

→ That is the whole point of this section. You will understand the whole process of how YOLO performs object detection and how to get image (B) from image (A).”

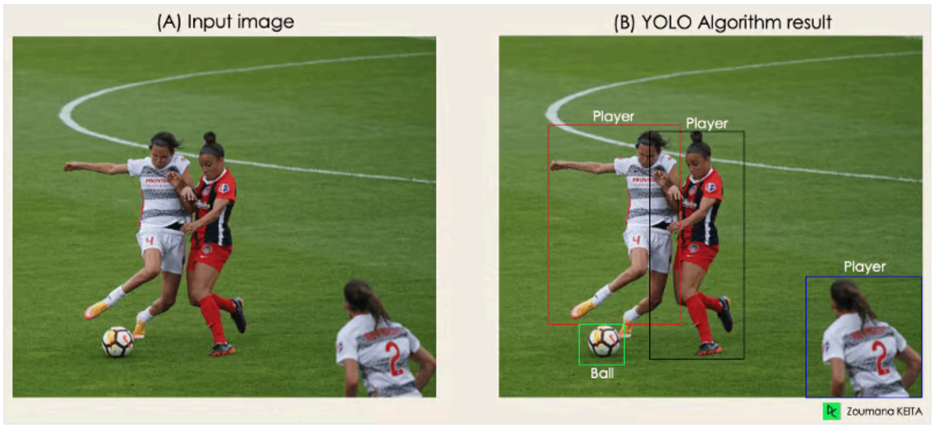


Image by Author

The algorithm works based on the following four approaches:

- Residual blocks
- Bounding box regression
- Intersection Over Unions or IOU for short
- Non-Maximum Suppression.

Let’s have a closer look at each one of them.

1. Residual blocks

This first step starts by dividing the original image (A) into NxN grid cells of equal shape, where N, in our case, is 4, as shown in the image on the right. Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, along with the probability/confidence value.

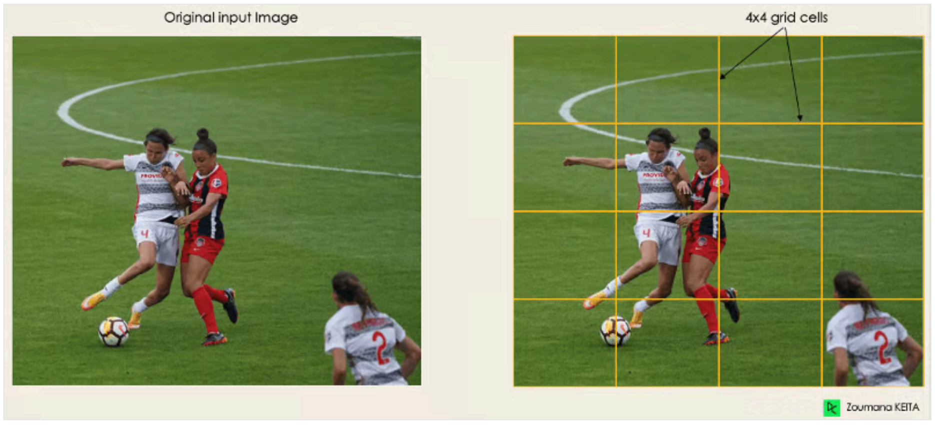


Image by Author

2. Bounding box regression

The next step is to determine the bounding boxes corresponding to rectangles, highlighting all the objects in the image. We can have as many bounding boxes as there are objects within a given image.

YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

`Y = [pc, bx, by, bh, bw, c1, c2]`

This is especially important during the training phase of the model.

- `pc` corresponds to the probability score of the grid containing an object. For instance, all the grids in red will have a probability score higher than zero. The image on the right is the simplified version since the probability of each yellow cell is zero (insignificant).

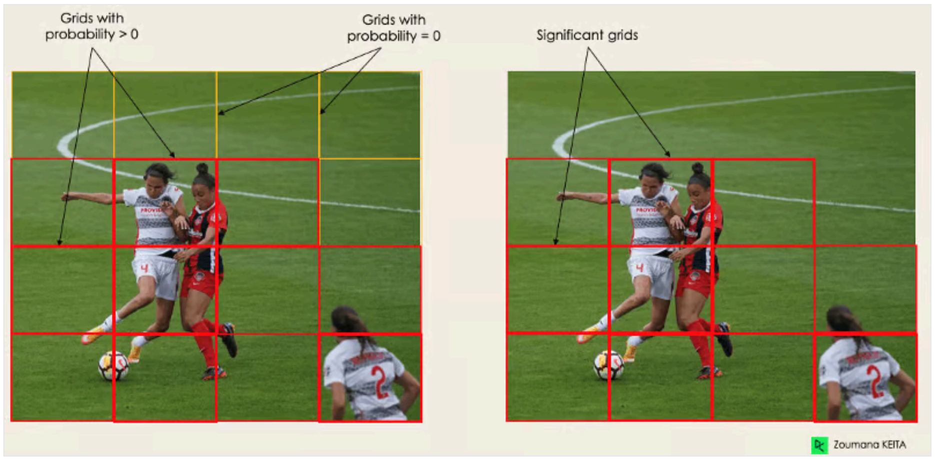


Image by Author

- `bx` , `by` are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell.
- `bh` , `bw` correspond to the height and the width of the bounding box with respect to the enveloping grid cell.
- `c1` and `c2` correspond to the two classes, Player and Ball. We can have as many classes as your use case requires.

To understand, let's pay closer attention to the player on the bottom right.

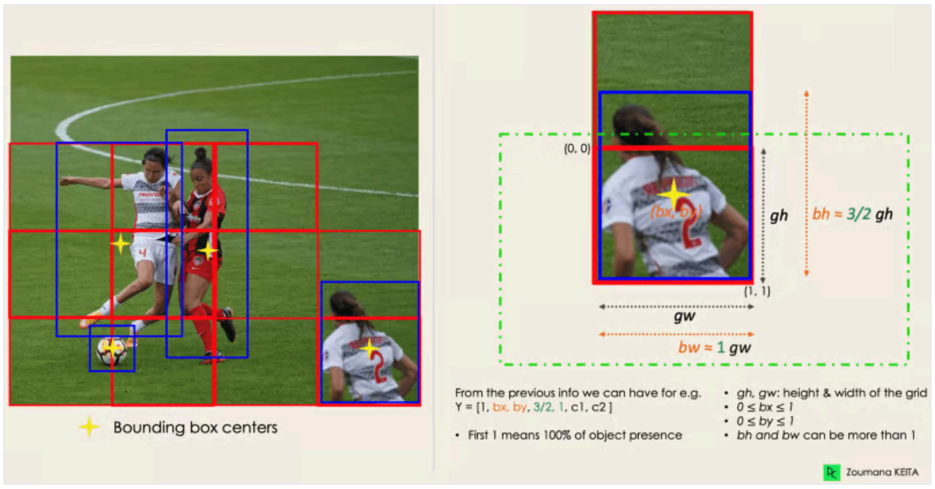

Bounding box centers
(0, 0)
(bx, by)
gh
bh ≈ 3/2 gh
(1, 1)
gw
bw ≈ 1 gw
From the previous info we can have for e.g.
Y = [1, bx, by, 3/2, 1, c1, c2]
• First 1 means 100% of object presence
• gh, gw: height & width of the grid
• 0 ≤ bx ≤ 1
• 0 ≤ by ≤ 1
• bh and bw can be more than 1
Zoumana KEITA

Image by Author

3. Intersection Over Unions or IOU

Most of the time, a single object in an image can have multiple grid box candidates for prediction, even though not all are relevant. The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant. Here is the logic behind it:

- The user defines its IOU selection threshold, which can be, for instance, 0.5.
- Then, YOLO computes the IOU of each grid cell, which is the Intersection area divided by the Union Area.
- Finally, it ignores the prediction of the grid cells having an IOU ≤ threshold and considers those with an IOU > threshold.

Below is an illustration of applying the grid selection process to the bottom left object. We can observe that the object originally had two grid candidates, and only "Grid 2" was selected at the end.